Computational Fabrication

CS 491 and 591
Professor: Leah Buechley
https://handandmachine.cs.unm.edu/classes/Computational_Fabrication

Calendar check in:

Data Physicalization Assignment

GCODE Assignment

Final Project Proposal Assignment

CAD CAM

CAD CAM

computer aided design

CAD CAM

computer aided manufacturing

3D Printing Workflow

CAD: Rhino, Grasshopper, and Python: design your geometry.

CAM part 1: Cura (or other "slicer"): translate geometry into machine readable (g-code) file by slicing it layer by layer and generating a tool path for each layer.

CAM part 2: Transfer the g-code file to the 3D printer. 3D printer interprets the g-code, (follows the tool path) and generates your artifact.

G-Code

Machine Code

https://www.autodesk.com/products/fusion-360/blog/computer-aided-manufacturing-beginners/

G-Code Overview

The language of machines; the code that tells the 3D printer (or other machine) what to do.

G-Code file: a series of simple commands that are interpreted line by line by the machine. Each line is one integrated command.

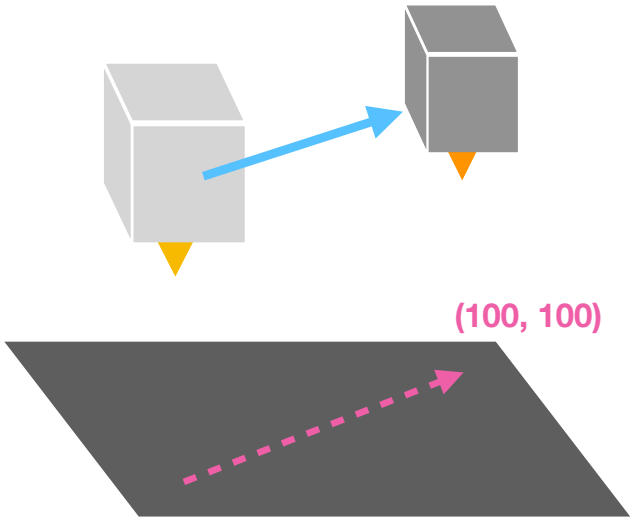
Basic elements of control:

- Movement of print head in x,y,z
- Extrusion of material (in one dimension)
- Temperature of bed and extruder ("hot end")

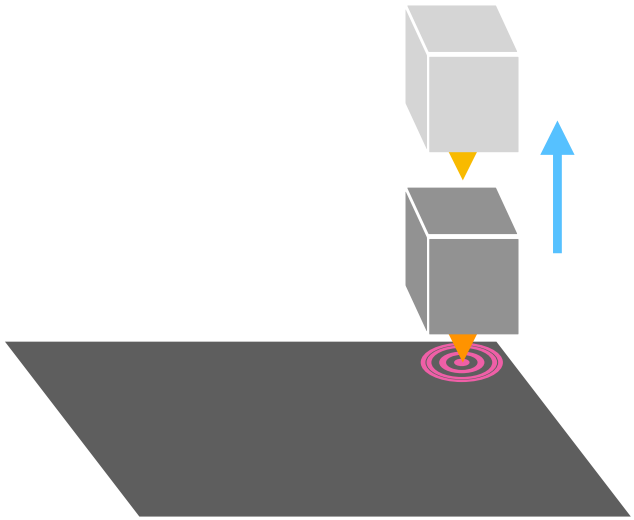
Command reference: https://reprap.org/wiki/G-code

Movement (mm) G1 or G01

```
G1 X100 Y100
```



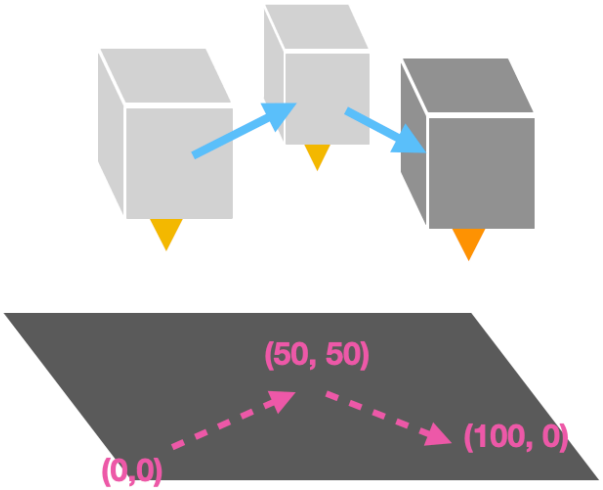

```
G1 Z1
```



Absolute (G90) vs. Relative (G91) Mode

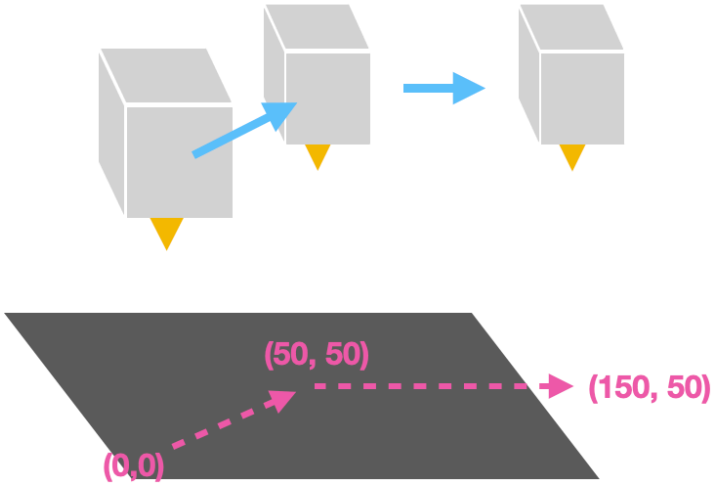
```
G90
G01 X50  Y50
G01 X100 Y0
```

(Absolute coordinates)



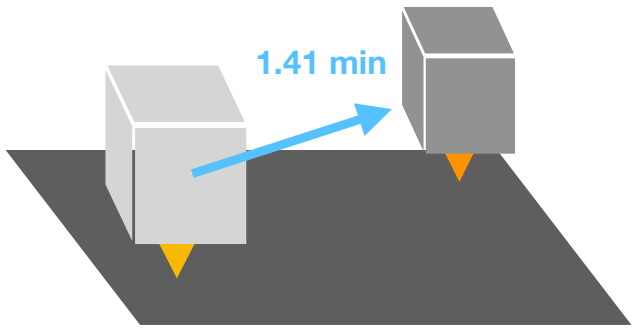

```
G91
G01 X50  Y50
G01 X100 Y0
```

(Relative coordinates)



Speed, AKA “Feedrate” (mm/minute) F

```
G1 X100 Y100 F100
```

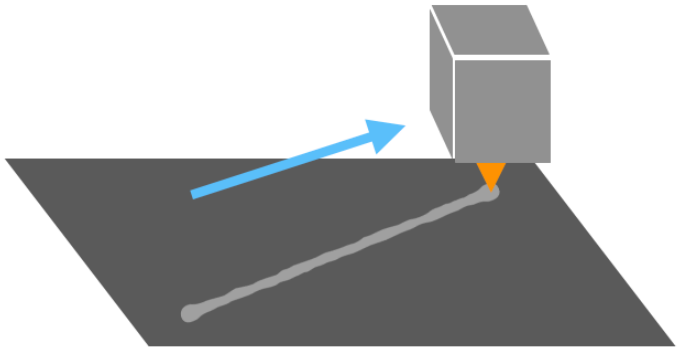


The speed of the print head as it moves from one point to another.

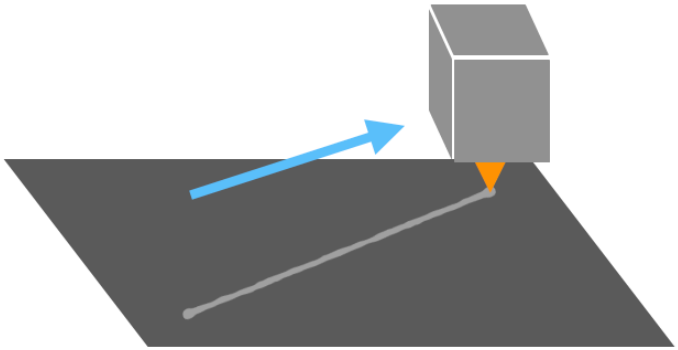
F1000 good starter speed

Extrusion (mm) E

```
G01 X100 Y100 E7
```



```
G01 X100 Y100 E3
```

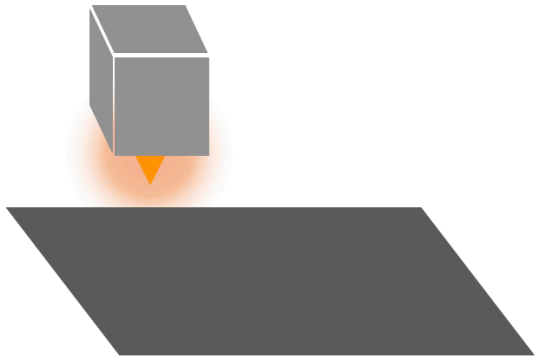


The amount of filament to extrude in mm across specified path.

Temperature

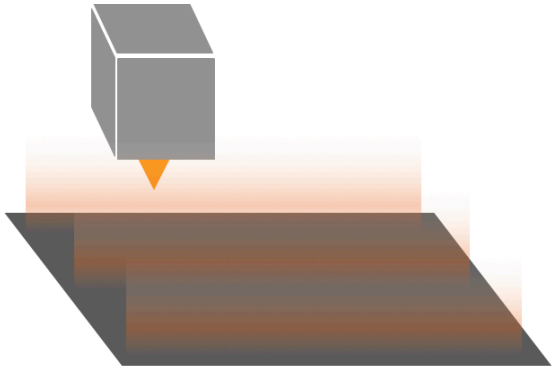
```
M104 S215
M109 S215
```

(Set hotend temperature, and wait)



```
M140 S60
M190 S60
```

(Set bed temperature, and wait)



M104: set extruder temperature, M140: set bed temperature

Other Useful Commands

G28 Home all axes

M0 Pause and wait for user interaction
G04 S100 Pause and wait for 100 ms, then continue

M84 Disable Motors

; Comments are anything on a line that follow a semi-colon

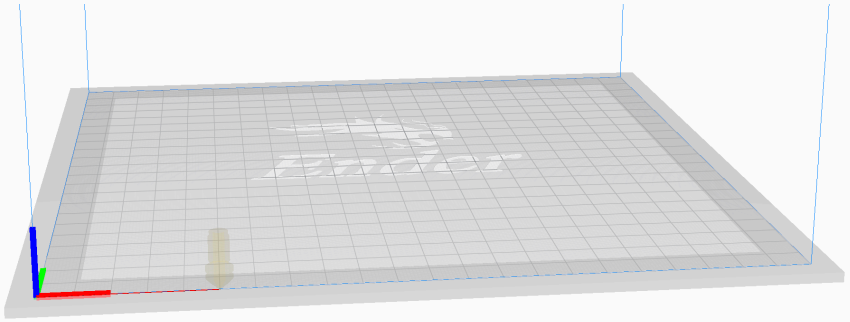
An Example File: Bed Leveling

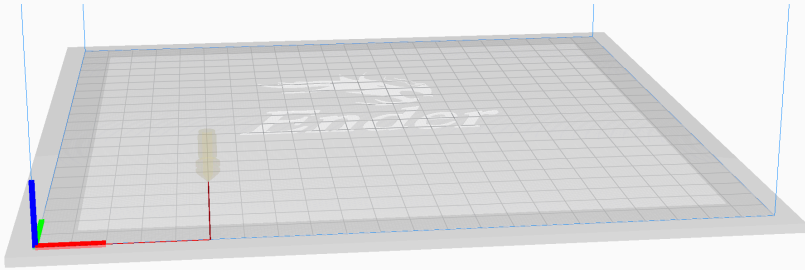
```
G90 ; Absolute mode for position

G28 ; Home all axis
G1 Z5 ; Lift Z axis
G1 X32 Y36 F3000; Move to Position 1
G1 Z0 ; Move Z axis down
M0 ; Pause print

G1 Z5 ; Lift Z axis
G1 X32 Y206 F3000; Move to Position 2
G1 Z0 ; Move Z axis down
M0 ; Pause print

G1 Z5 ; Lift Z axis
G1 X202 Y206 F3000; Move to Position 3
G1 Z0
M0 ; Pause print

G1 Z5 ; Lift Z axis
G1 X202 Y36 F3000; Move to Position 4
G1 Z0
```

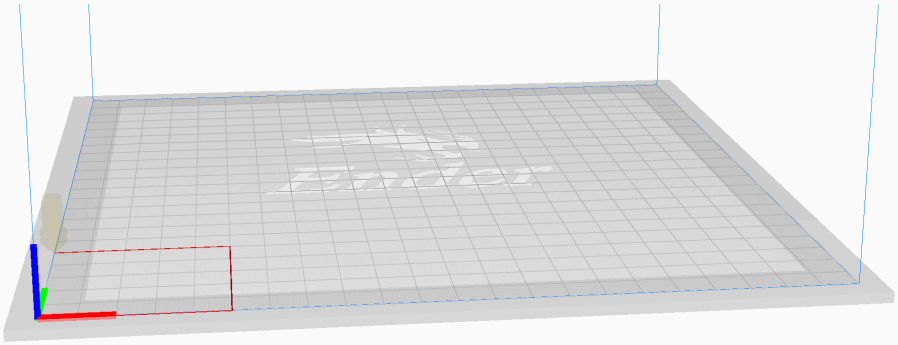
An Example File: Draw a Square

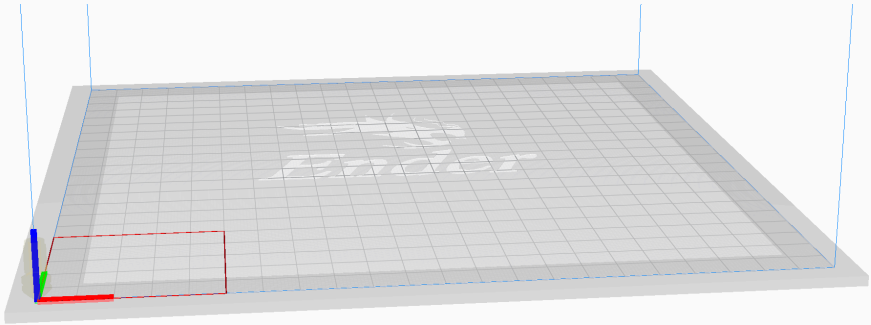
```
G92 E0    ; Reset Extruder
G28       ; Home all axes
M190 S60  ; Set bed temperature and wait
M109 S205 ; Set extruder temperature and wait
G1 F1000  ; Set feedrate (speed) to 1000 mm/s
G91       ; Relative mode for position

; Draw a square
G1 X50.0 Y0.0 Z0.0 E5.0
G1 X0.0 Y50.0 Z0.0 E5.0
G1 X-50.0 Y0.0 Z0.0 E5.0
G1 X-0.0 Y-50.0 Z0.0 E5.0
```

Preview in Cura









RepRap Wiki

Main Page
Build a RepRap
Glossary
Reference

Participation
Recent Changes
Get a Wiki account
Create a new page
Policy

Community
RepRap Forum
RepRap IRC
Development Index
RepRap User Groups

Page | Discussion

G-code

English • العربية • български • català • čeština • Deutsch • Ελληνικά • español • فارسی • українська • 中文（中国大陆） • 中文（台灣） • עברית • azərbaycanca •

This page tries to describe the flavour of **G-codes** that the RepRap firmwares u
NIST RS274NGC G-code standard, so RepRap firmwares are quite usable fo

There are a few different ways to prepare G-code for a printer. One method wou
output the G-code required to print each layer. Slicers are the easiest way to go
generation is to use a lower level library like mecode. Libraries like mecode give
final option is to just write the G-code yourself. This may be the best choice if yc

As many different firmwares exist and their developers tend to implement new f
specific codes developed over the years. This particular page is the master pag
The rule is: **add your new code here, then implement it**.

Unfortunately human nature being what it is, the best procedures aren't always
on this page (later than the original use of a code), are deprecated and should b
Note that the key date is appearance here, not date of implementation.

https://reprap.org/wiki/G-code

Depth of G-Code Capabilities

Beyond 3D printers: CNC milling machines, lathes, etc.

Low level control of mechanical components of machines like motors

Machine set up and configuration as well as control

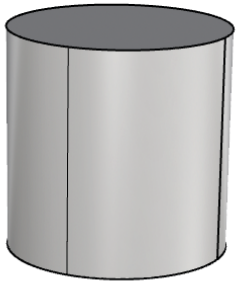
Some ability to include variables and more complex code structures

We will focus on simple 3D printing toolpaths

questions?

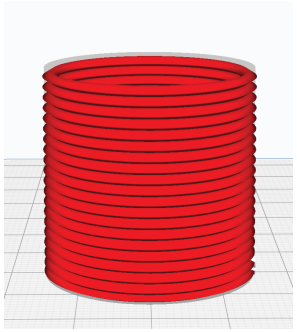
Why generate your own G-Code?

Traditional Workflow



CAD model

slicing software



toolpath

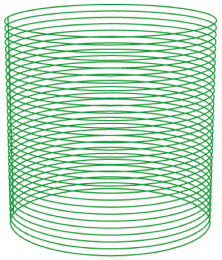
3D print

Alternative Workflow

```
t = ExtruderTurtle()

for i in range (layers):
  for j in range (360):
    t.forward(1)
    t.right(1)
  t.lift(layer_height)
```

program

program output

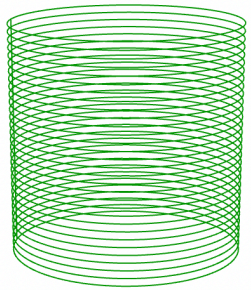


toolpath

3D print

Direct Machine Control

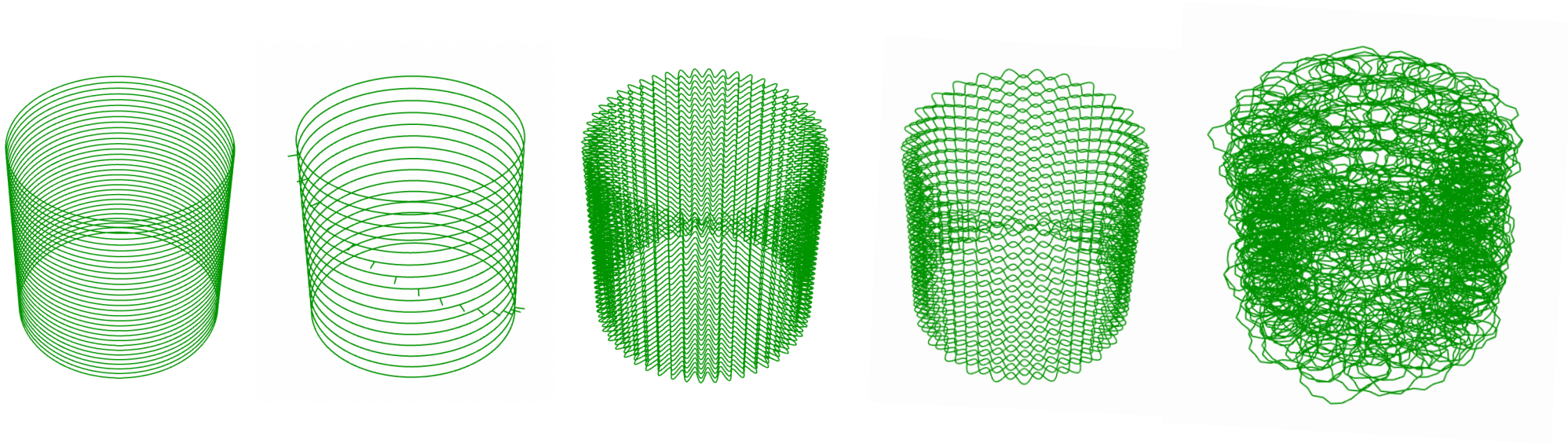
```
elif (printer=="Eazao" or printer=="eazao"):
    if(self.out_file):
        self.initseq_filename = os.path.join
    self.nozzle = 1.5
    self.extrude_width = 2.2
    self.layer_height = 1.0
    self.extrude_rate = 1.0 #mm extruded/mm
    self.speed = 1500 #mm/minute
    self.printer = "eazao"
    self.resolution = 1.0
    self.x_size = 150
    self.y_size = 150
```



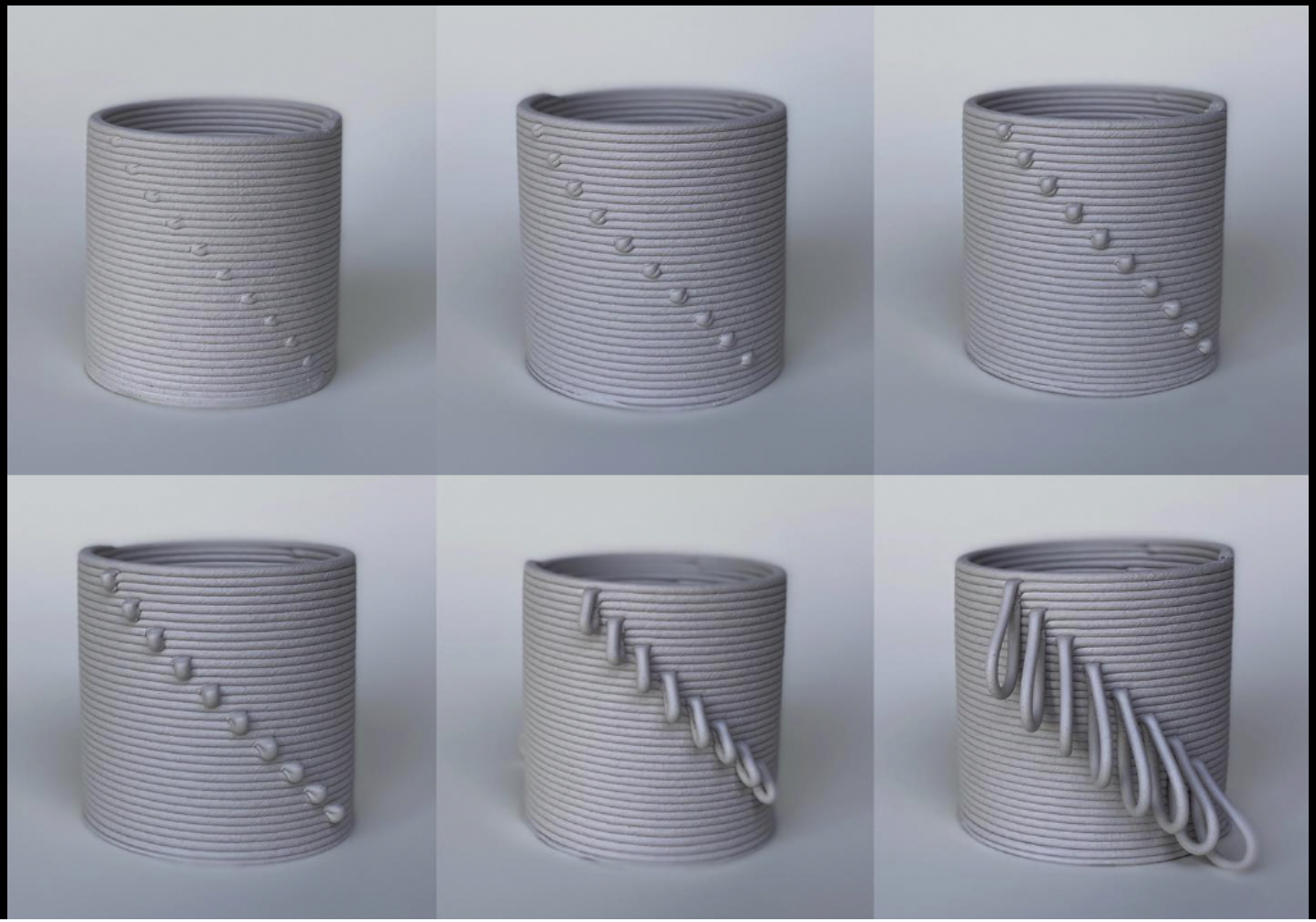
- Skip CAD design and slicing software
- Design a toolpath & object directly by writing code
- Output = 3D printer tool path
- Toolpath determines geometry
- Toolpath also determines surface properties
- Fine-grained control over printer behavior

Direct Machine Control

- Toolpaths you can't create with slicers:

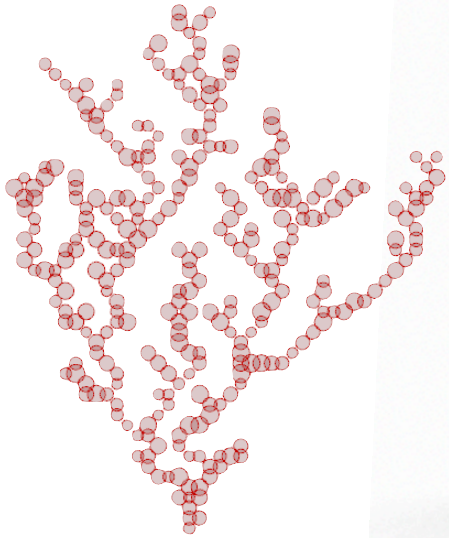


Examples



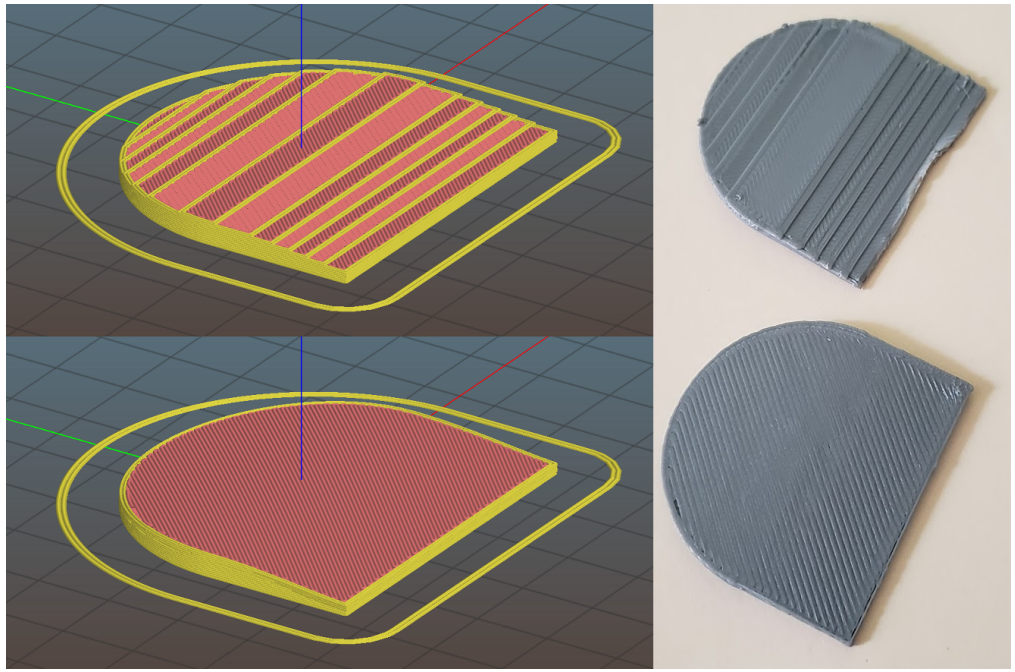
Examples

Example: Matrix to surface pattern



More examples

Nonplanar (angled) paths

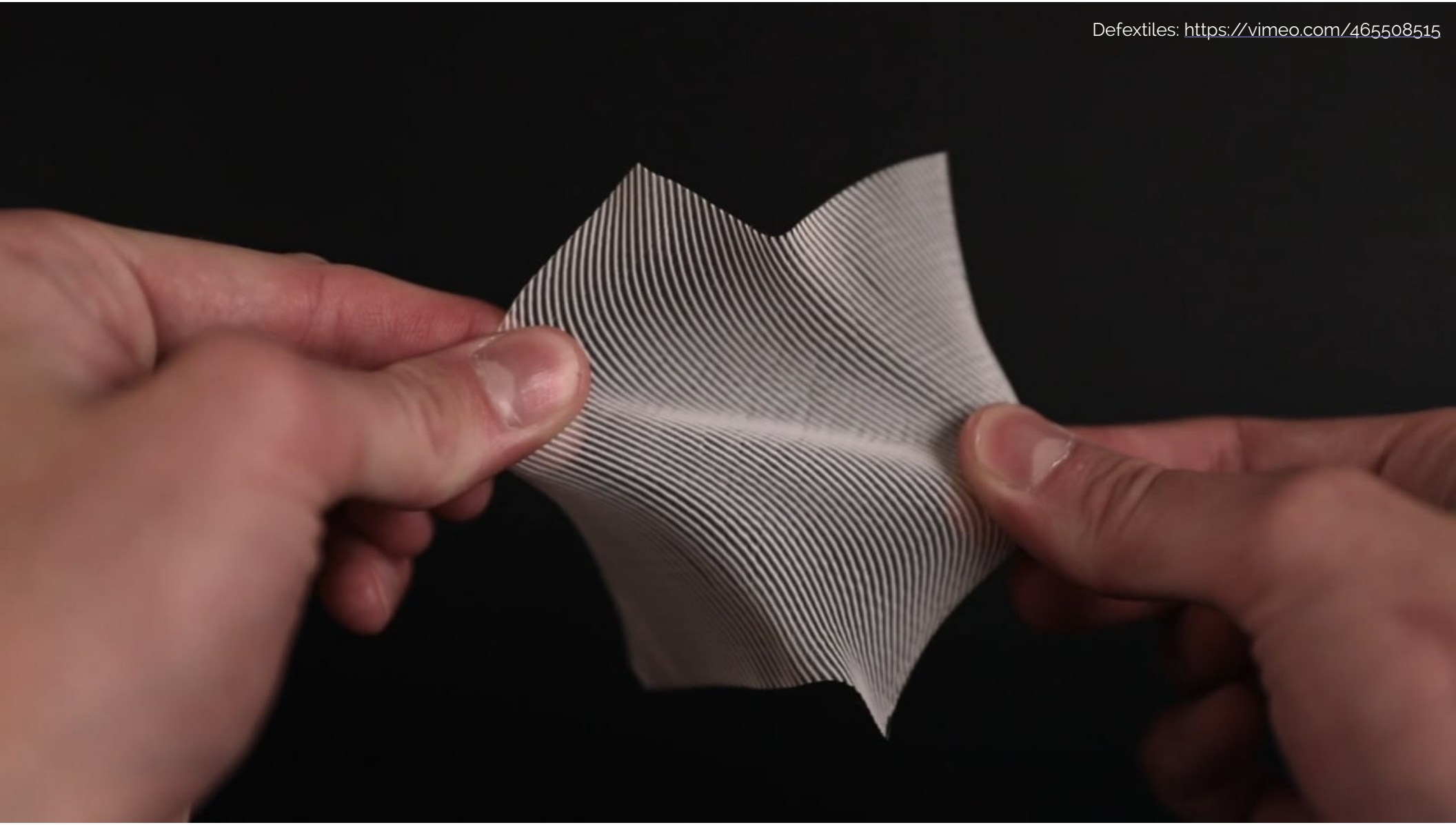


Gentle slopes tend not to come out very well when sliced into horizontal planes.

Nonplanar slicing can be used to avoid the “stair stepping effect” for gently sloped surfaces.

Instead of horizontal planes, the solid is sliced into slightly curved sheets.

Nonplanar Slicing. (2020, September 14). In *Appropedia*. https://www.appropedia.org/Nonplanar_Slicing


Defextiles: https://vimeo.com/465508515

A Different Kind of Example:
Modify Existing G-Code Files
Experiments by Franklin Pezutti-Dyer

“Pug buddy” test print

Alteration of the “Pug buddy” test print in which more and more randomness is added for layers with higher Z-values

Same premise, but with less randomness,
and a more gradual increase in randomness

Another example, in which randomness is only added to the right side
(Only perturb coordinates with an X-value above the average X-value)

A different transformation: “twist” the print by rotating X and Y coordinates about a vertical axis, increasing the rotation amount for layers with greater Z-values

Some failed experiments:

NAILED IT!

Hand and Machine “ExtruderTurtle” Library for generating GCODE

Turtle Geometry & LOGO

LOGO

An embodied approach to geometry

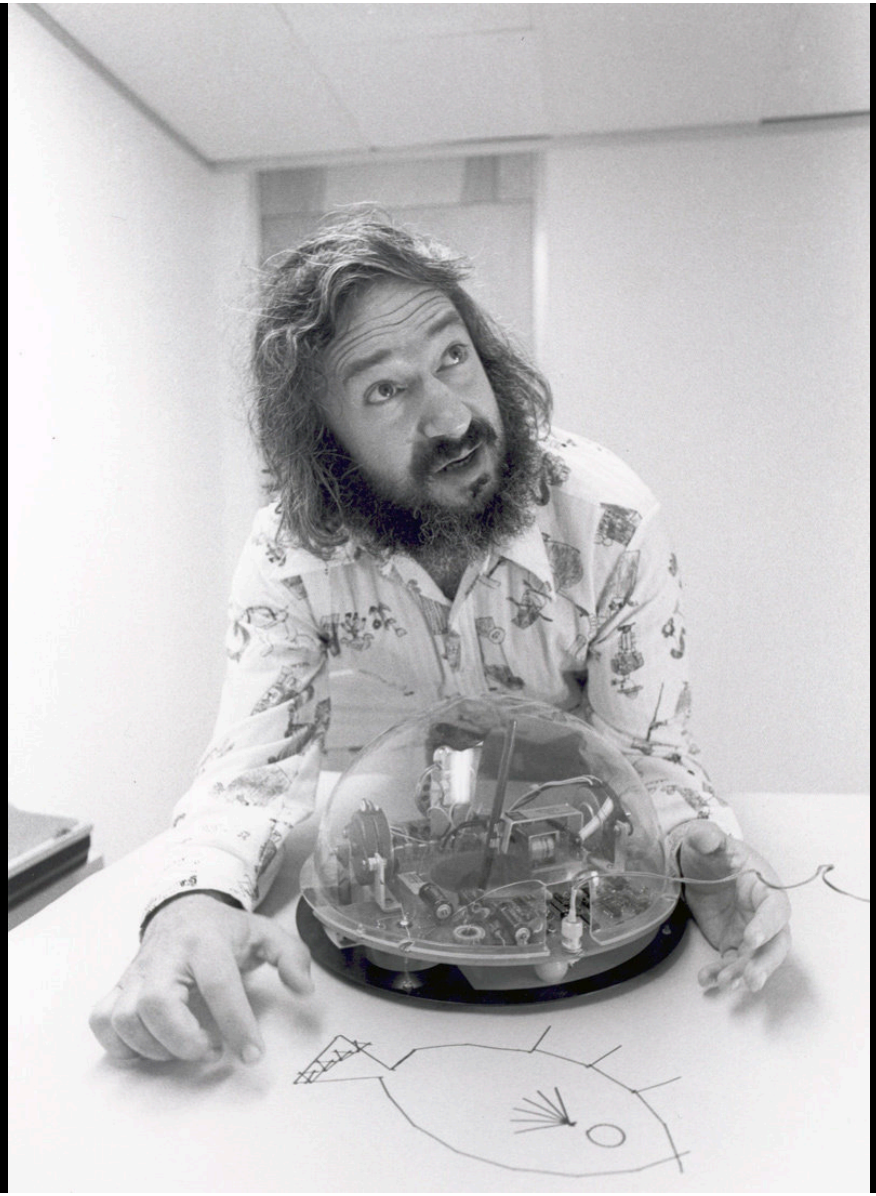
You direct a "turtle" to move around giving it simple directions.

Developed in 1967 by
Seymour Papert, Cynthia Solomon, and Wally Feurzeig

A language designed to enable children to explore
mathematics and computers

Turtle robot in 1969

Mindstorms published in 1980
Turtle Geometry published in 1981



Seymour Papert and students, 1969

Traditional LOGO commands

forward(amount): move forward the given amount
back(amount): move backward the given amount

right (angle): turn right the given angle
left (angle): turn left the given angle

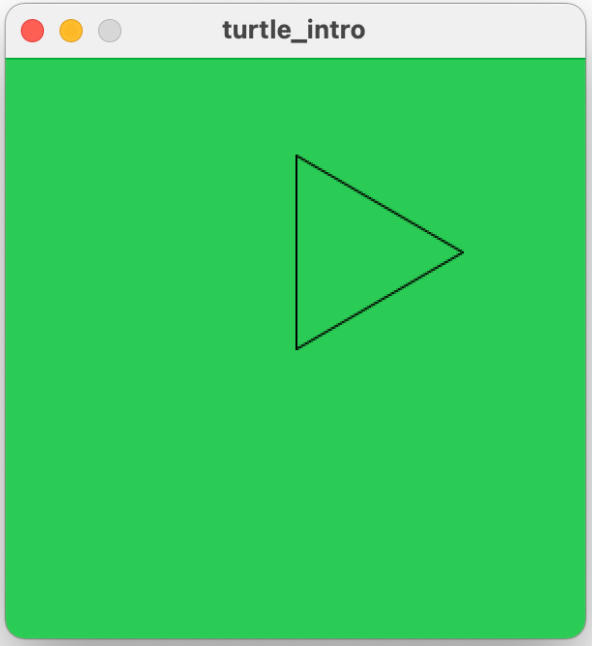
penup (): stop drawing
pendown(): start drawing

Turtle Demonstration

Using Processing & a custom Turtle library

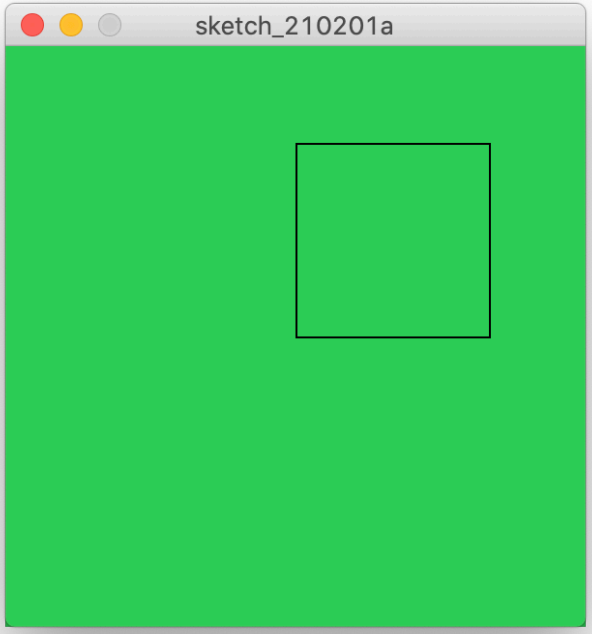
Drawing a Triangle

```
import Turtle.*;
Turtle t;

void setup() {
  size(300,300);
  background(100,200,100);
  t = new Turtle (this);
}

void draw() {
  t.forward(100);
  t.right(120);
}
```



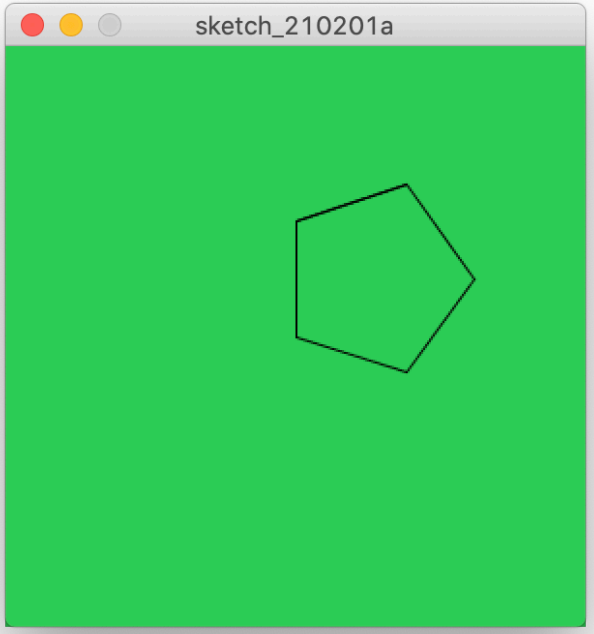
A Square

```
import Turtle.*;
Turtle t;

void setup() {
  size(300,300);
  background(100,200,100);
  t = new Turtle (this);
  frameRate(3);
}

void draw() {
  t.forward(100);
  t.right(90);
}
```



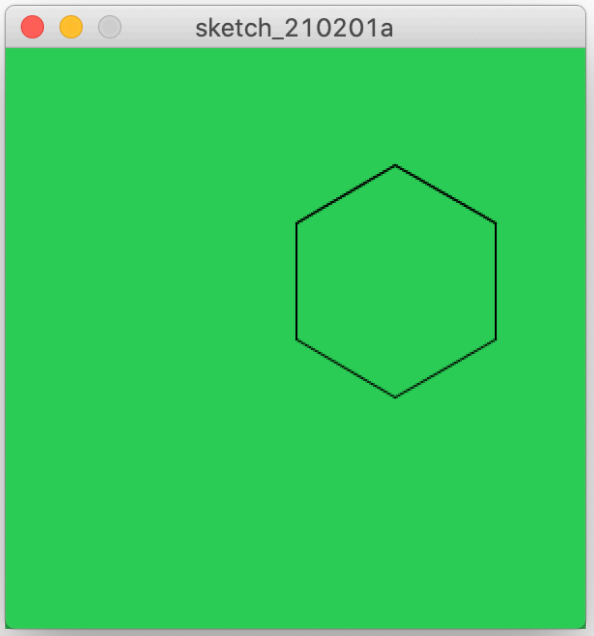
Polygons

```
import Turtle.*;
Turtle t;

void setup() {
  size(300,300);
  background(100,200,100);
  t = new Turtle(this);
  frameRate(3);
}

void draw() {
  polygonStep(60,72);
}

void polygonStep(int size, int angle) {
    t.forward(size);
    t.right(angle);
}
```



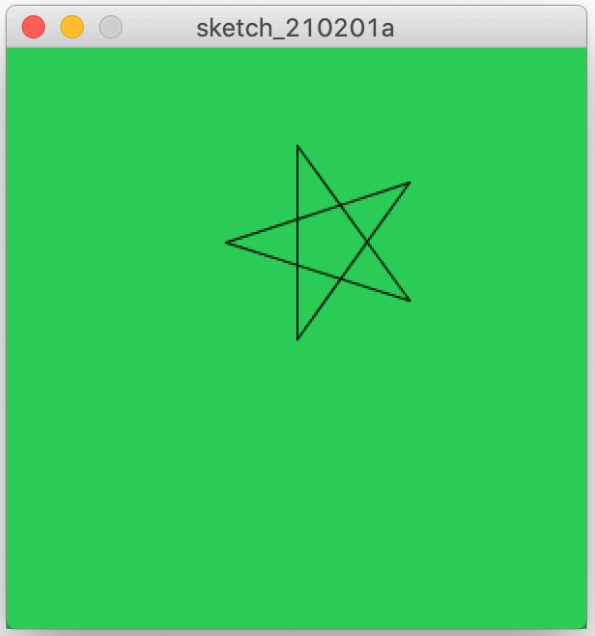
Polygons

```
import Turtle.*;
Turtle t;

void setup() {
  size(300,300);
  background(100,200,100);
  t = new Turtle(this);
  frameRate(3);
}

void draw() {
  polygonStep(60,60);
}

void polygonStep(int size, int angle) {
    t.forward(size);
    t.right(angle);
}
```



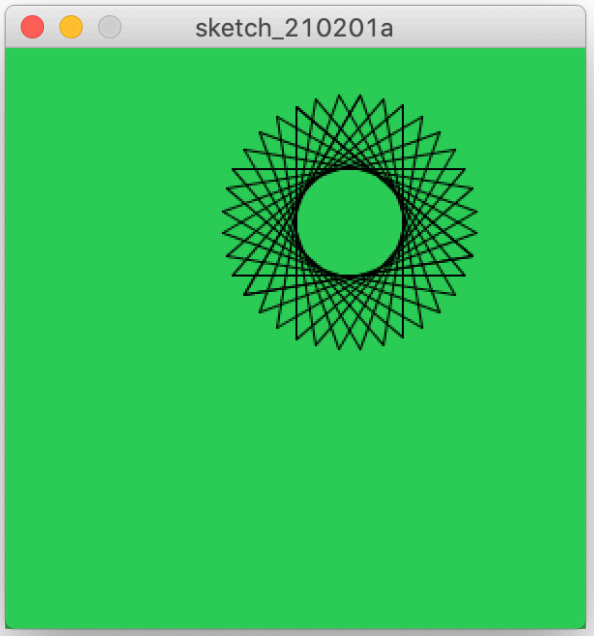
Polygons

```
import Turtle.*;
Turtle t;

void setup() {
  size(300,300);
  background(100,200,100);
  t = new Turtle(this);
  frameRate(3);
}

void draw() {
  polygonStep(100,144);
}

void polygonStep(int size, int angle) {
    t.forward(size);
    t.right(angle);
}
```



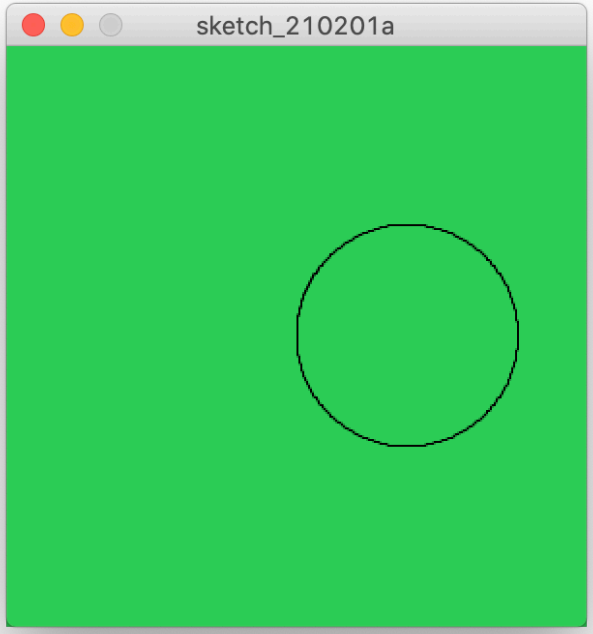
Polygons

```
import Turtle.*;
Turtle t;

void setup() {
  size(300,300);
  background(100,200,100);
  t = new Turtle(this);
  frameRate(3);
}

void draw() {
  polygonStep(100,130);
}

void polygonStep(int size, int angle) {
    t.forward(size);
    t.right(angle);
}
```



A Circle

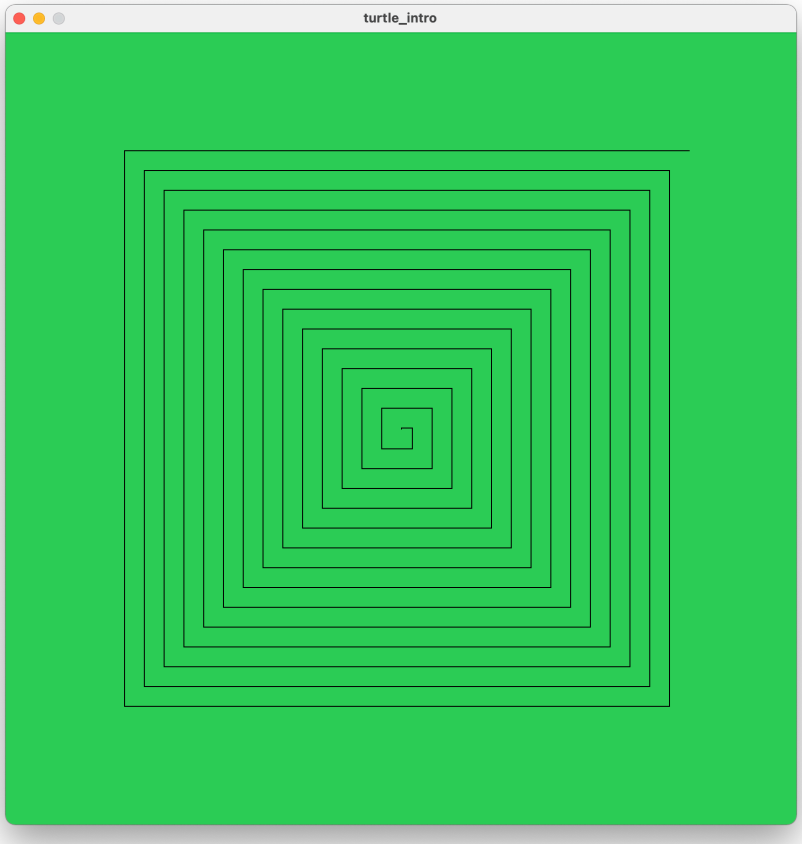
```
import Turtle.*;
Turtle t;

void setup() {
  size(300,300);
  background(100,200,100);
  t = new Turtle(this);
  frameRate(50);
}

void draw() {
  polygonStep(1,1);
}

void polygonStep(int size, int angle) {
    t.forward(size);
    t.right(angle);
}
```



Playing with parameters…

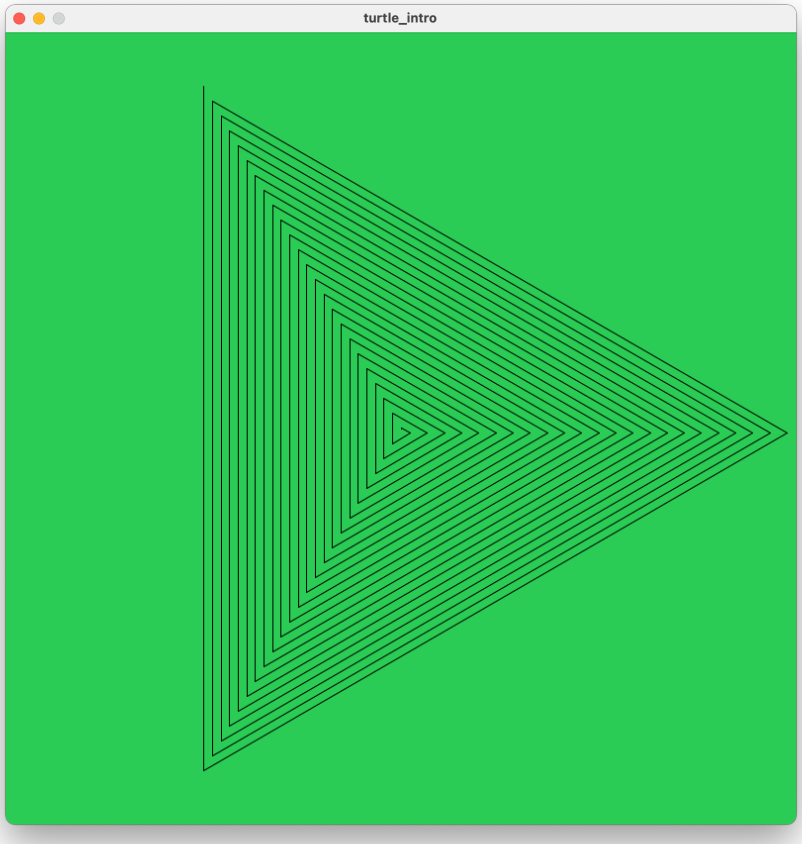
Changing Size at Each Step

```
import Turtle.*;
Turtle t;
int size, angle;

void setup() {
  size(300,300);
  background(100,200,100);
  t = new Turtle(this);
  frameRate(10);
  size = 1;
  angle = 90;
}

void draw() {
  polygonStep(size,angle);
  size = size + 10;
}

void polygonStep(int size, int angle) {
    t.forward(size);
    t.right(angle);
}
```



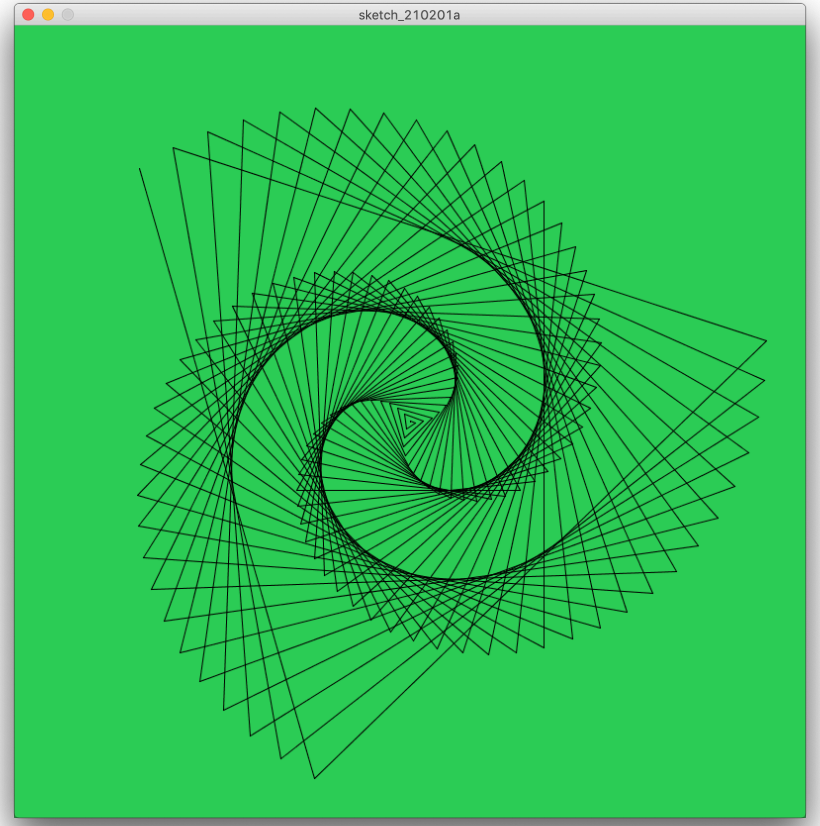
Changing Size

```
import Turtle.*;
Turtle t;
int size, angle;

void setup() {
  size(800,800);
  background(100,200,100);
  t = new Turtle(this);
  frameRate(10);
  size = 1;
  angle = 120;
}

void draw() {
  polygonStep(size,angle);
  size = size + 10;
}

void polygonStep(int size, int angle) {
    t.forward(size);
    t.right(angle);
}
```



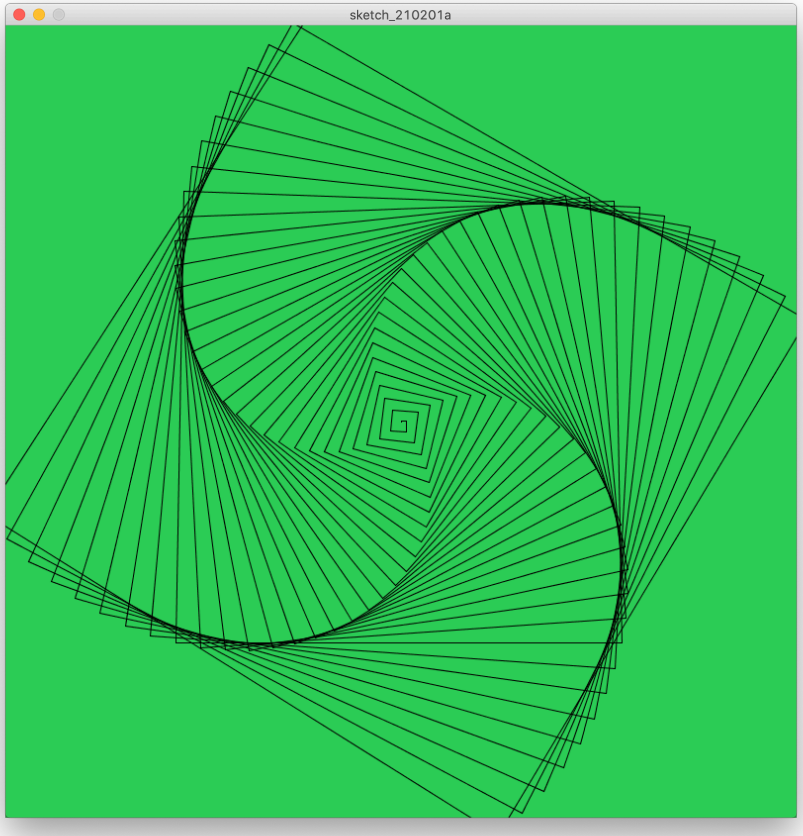
Changing Size & Starting Angle

```
import Turtle.*;
Turtle t;
int size, angle;

void setup() {
  size(800,800);
  background(100,200,100);
  t = new Turtle(this);
  frameRate(10);
  size = 1;
  angle = 118;
}

void draw() {
  polygonStep(size,angle);
  size = size + 5;
}

void polygonStep(int size, int angle) {
    t.forward(size);
    t.right(angle);
}
```



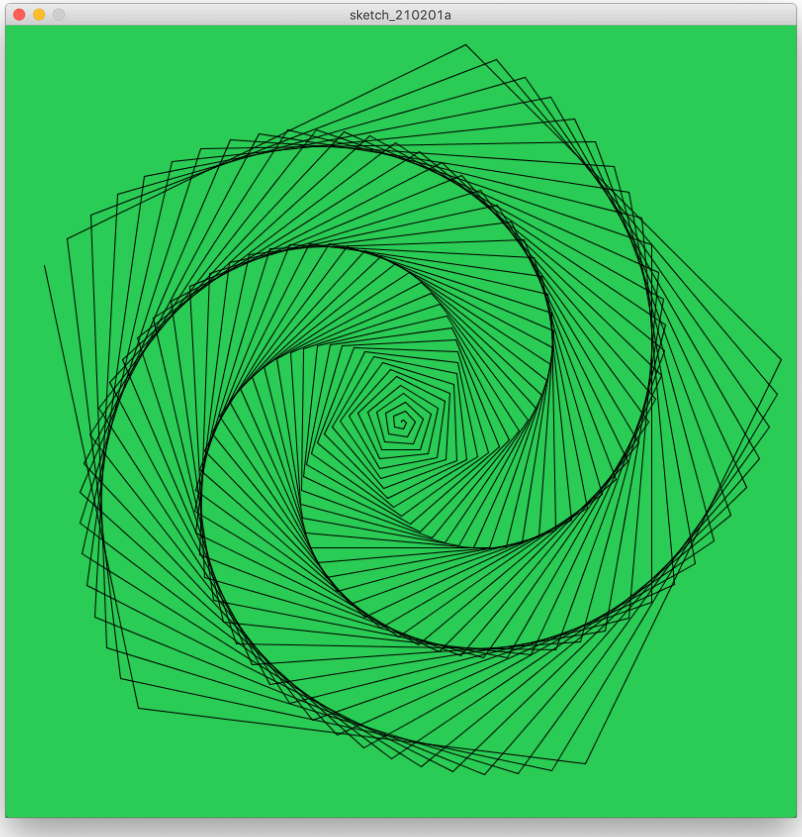
Changing Size & Starting Angle

```
import Turtle.*;
Turtle t;
int size, angle;

void setup() {
  size(800,800);
  background(100,200,100);
  t = new Turtle(this);
  frameRate(10);
  size = 1;
  angle = 91;
}

void draw() {
  polygonStep(size,angle);
  size = size + 5;
}

void polygonStep(int size, int angle) {
    t.forward(size);
    t.right(angle);
}
```



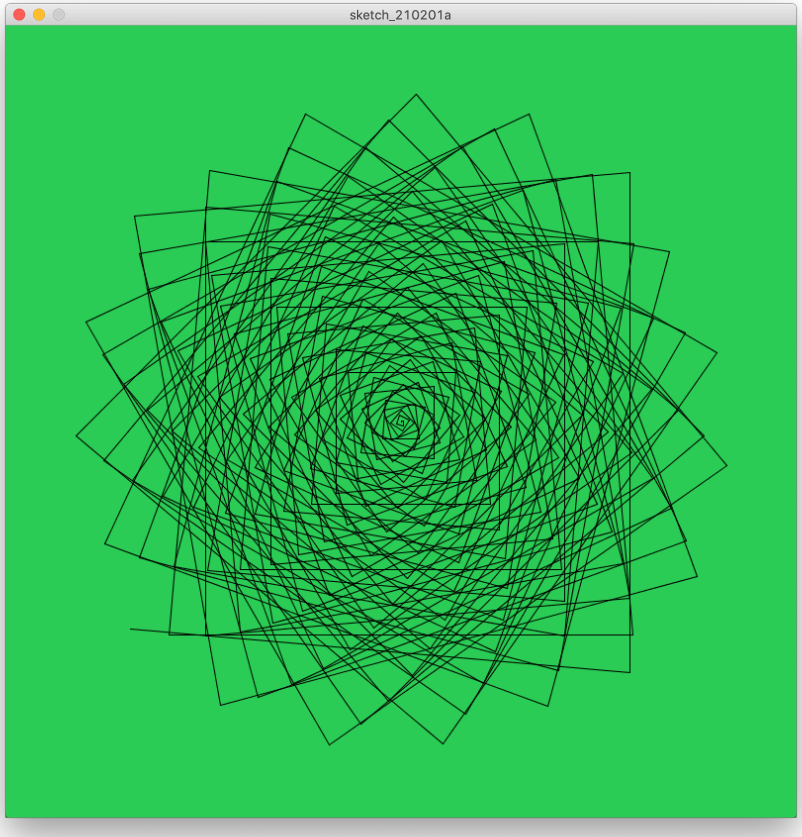
Changing Size & Starting Angle

```
import Turtle.*;
Turtle t;
int size, angle;

void setup() {
  size(800,800);
  background(100,200,100);
  t = new Turtle(this);
  frameRate(10);
  size = 1;
  angle = 73;
}

void draw() {
  polygonStep(size,angle);
  size = size + 2;
}

void polygonStep(int size, int angle) {
    t.forward(size);
    t.right(angle);
}
```



Changing Size & Starting Angle

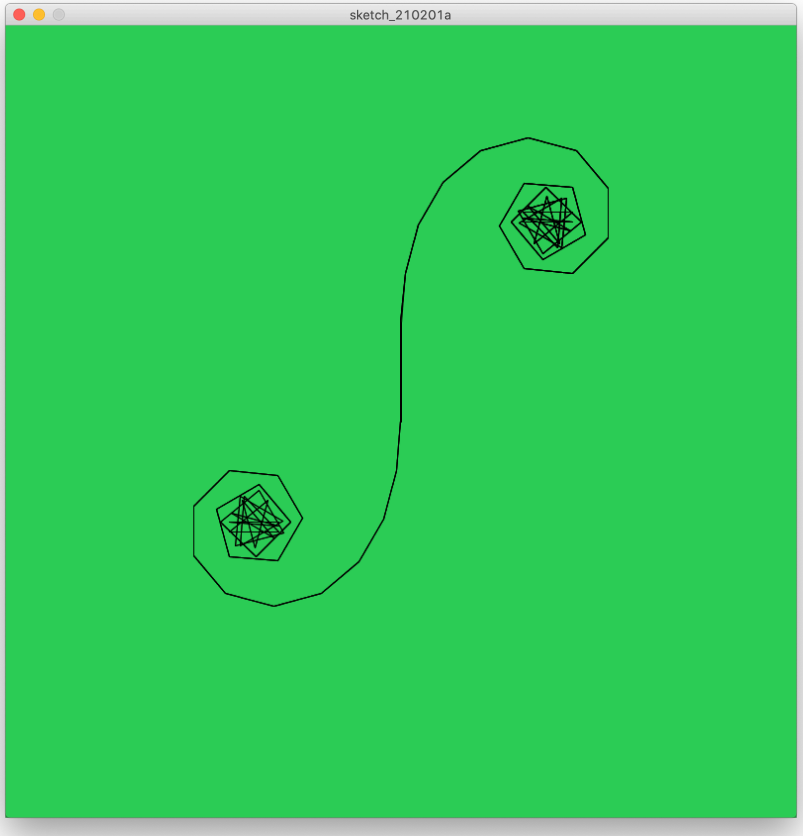
```
import Turtle.*;
Turtle t;
int size, angle;

void setup() {
  size(800,800);
  background(100,200,100);
  t = new Turtle(this);
  frameRate(10);
  size = 1;
  angle = 95;
}

void draw() {
  polygonStep(size,angle);
  size = size + 2;
}

void polygonStep(int size, int angle) {
    t.forward(size);
    t.right(angle);
}
```



Playing with parameters…

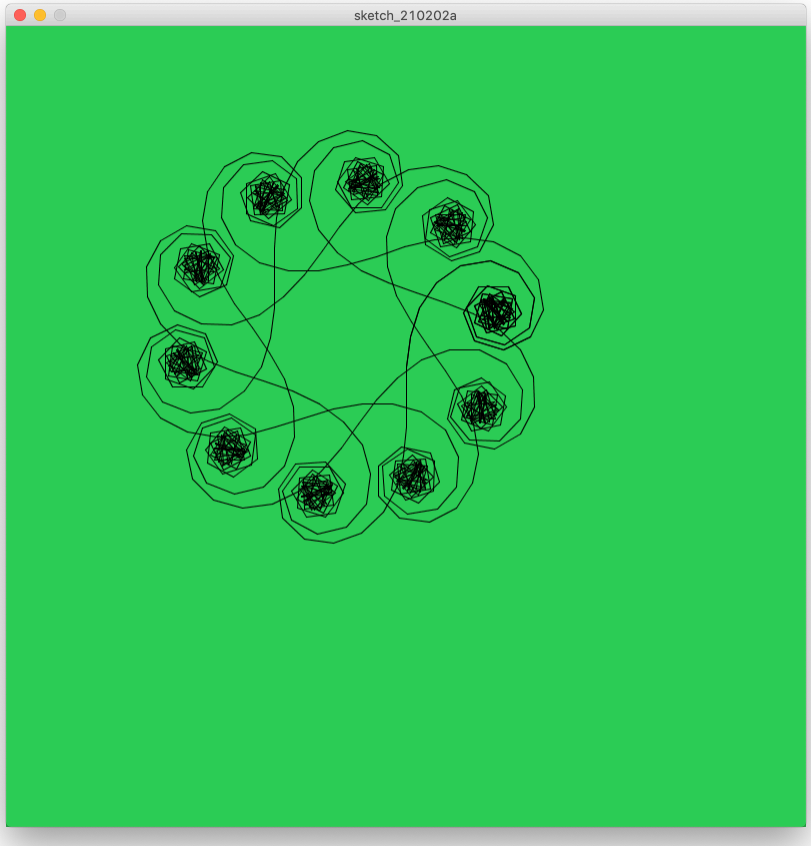
Changing Angle (Increment)

```
import Turtle.*;
Turtle t;
int size, angle;

void setup() {
  size(800,800);
  background(100,200,100);
  t = new Turtle(this);
  size = 50;
  angle = 0;
}

void draw() {
  polygonStep(size,angle);
  angle= angle + 5;
}

void polygonStep(int size, int angle) {
    t.forward(size);
    t.right(angle);
}
```



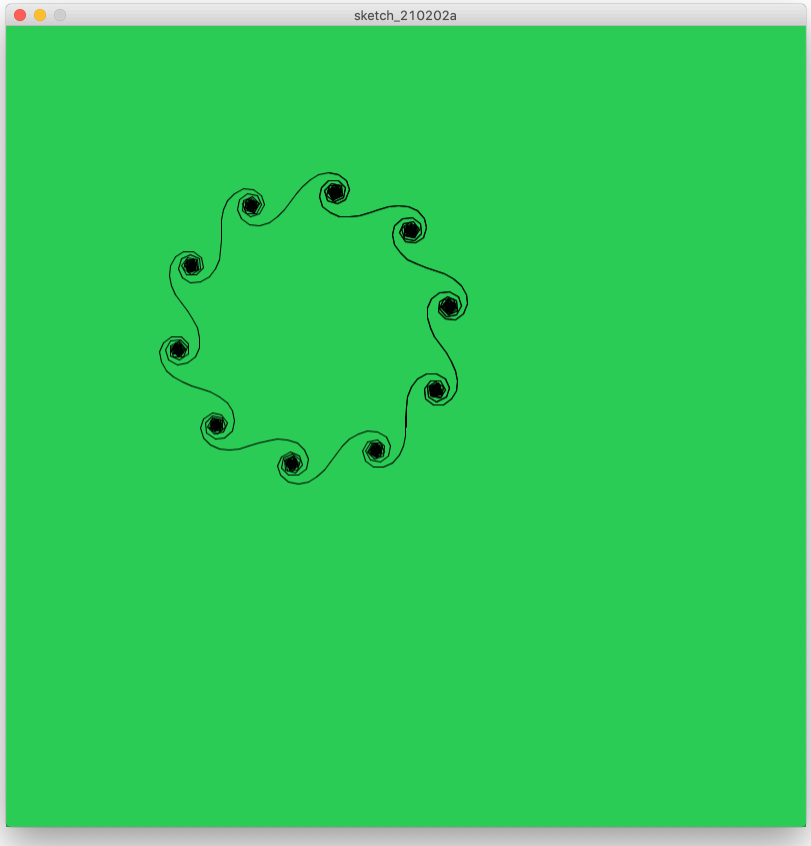
+ Changing (Starting) Angle

```
import Turtle.*;
Turtle t;
int size, angle;

void setup() {
  size(800,800);
  background(100,200,100);
  t = new Turtle(this);
  size = 30;
  angle = 1;
}

void draw() {
  polygonStep(size,angle);
  angle= angle + 5;
}

void polygonStep(int size, int angle) {
    t.forward(size);
    t.right(angle);
}
```



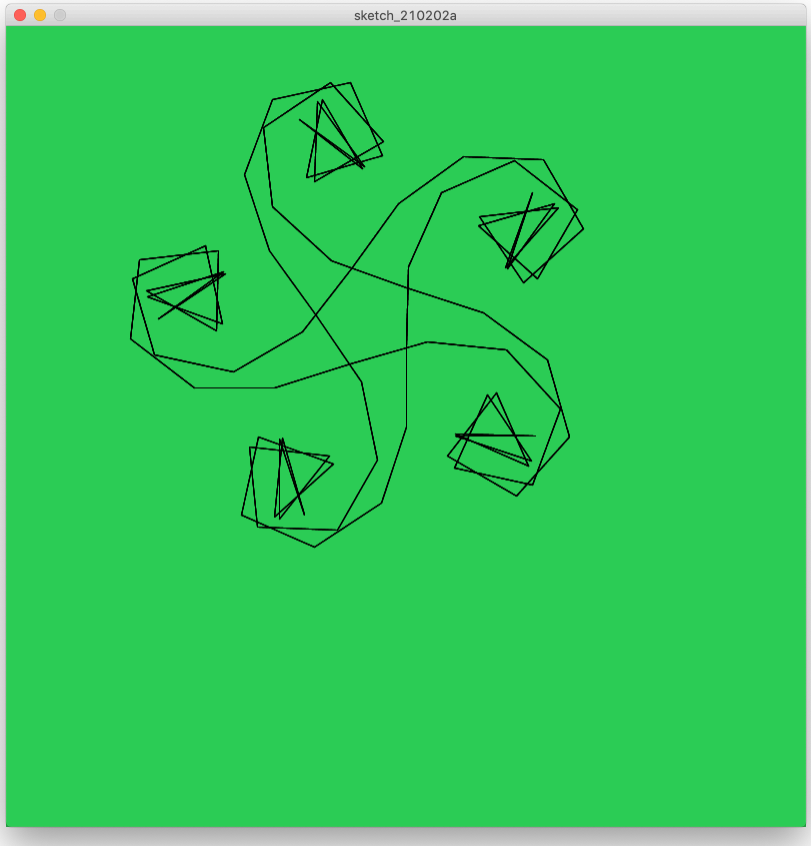
+ Changing (Starting) Angle

```
import Turtle.*;
Turtle t;
int size, angle;

void setup() {
  size(800,800);
  background(100,200,100);
  t = new Turtle(this);
  size = 10;
  angle = 2;
}

void draw() {
  polygonStep(size,angle);
  angle= angle + 5;
}

void polygonStep(int size, int angle) {
    t.forward(size);
    t.right(angle);
}
```



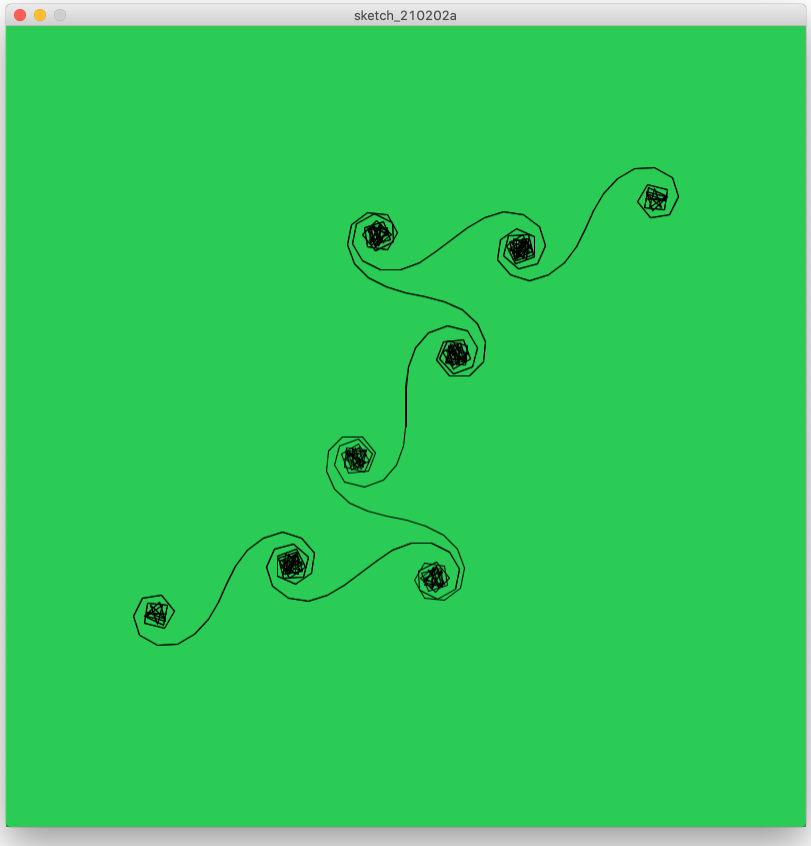
Changing Angles

```
import Turtle.*;
Turtle t;
int size, angle;

void setup() {
  size(800,800);
  background(100,200,100);
  t = new Turtle(this);
  size = 80;
  angle = 2;
}

void draw() {
  polygonStep(size,angle);
  angle= angle + 20;
}

void polygonStep(int size, int angle) {
    t.forward(size);
    t.right(angle);
}
```



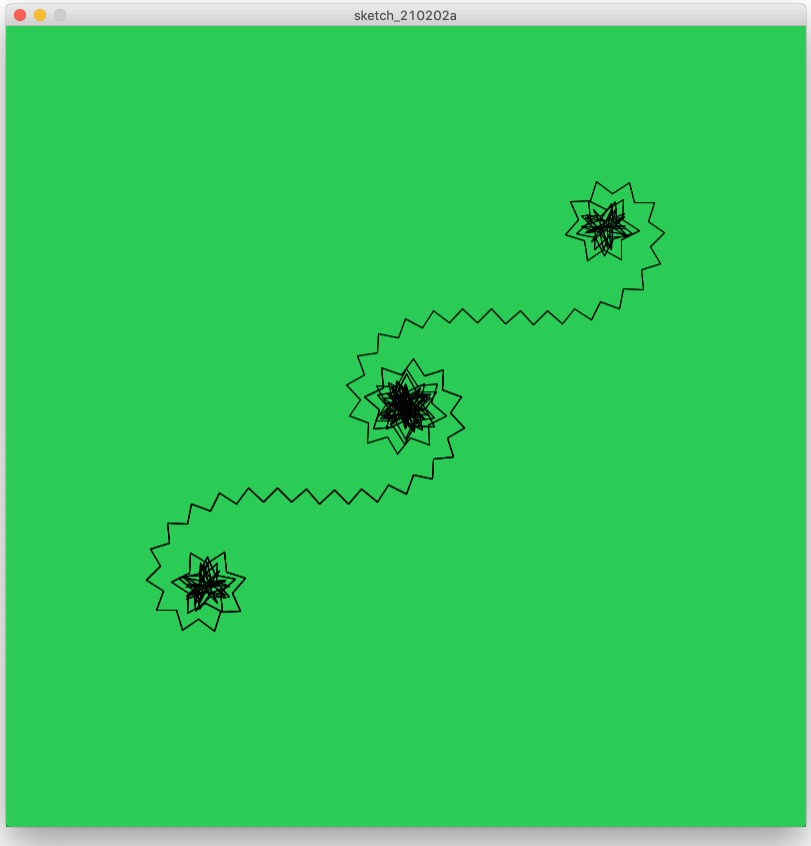
Changing Angles

```
import Turtle.*;
Turtle t;
int size, angle;

void setup() {
  size(800,800);
  background(100,200,100);
  t = new Turtle(this);
  size = 20;
  angle = 0;
}

void draw() {
  polygonStep(size,angle);
  angle= angle + 7;
}

void polygonStep(int size, int angle) {
    t.forward(size);
    t.right(angle);
}
```



Changing Angles

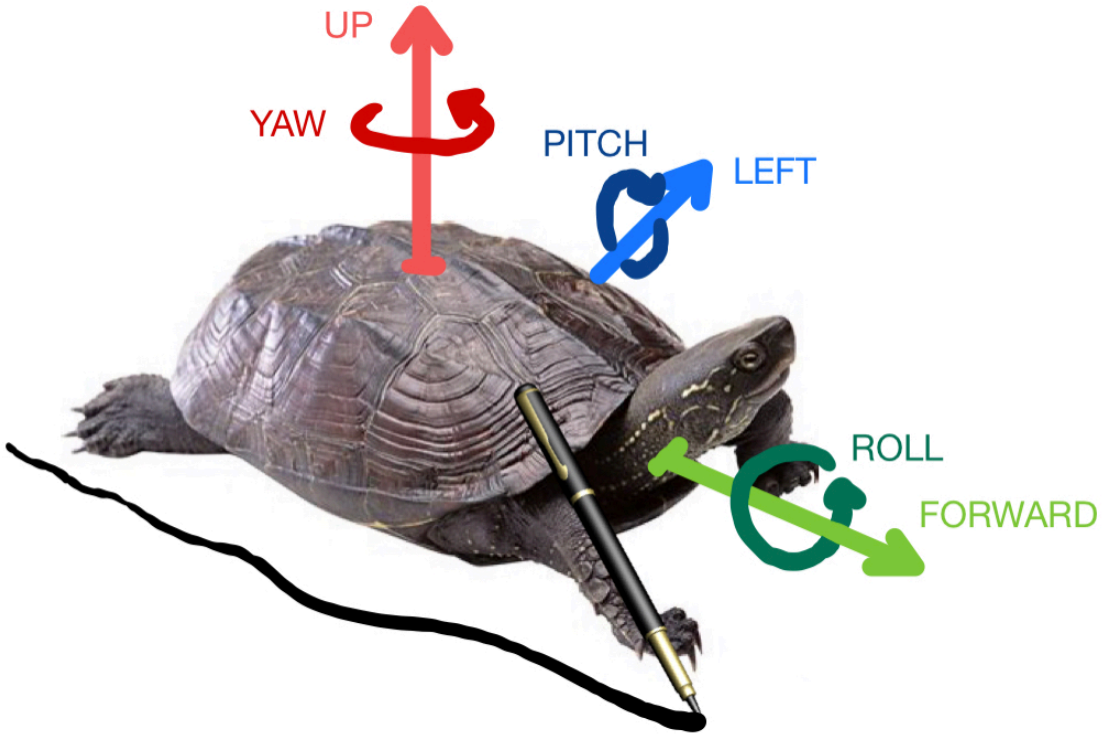
```
import Turtle.*;
Turtle t;
int size, angle;

void setup() {
  size(800,800);
  background(100,200,100);
  t = new Turtle(this);
  size = 20;
  angle = 0;
}

void draw() {
  polygonStep(size,angle);
  angle= angle + 179;
}

void polygonStep(int size, int angle) {
    t.forward(size);
    t.right(angle);
}
```



different way of making sense of space

Hand and Machine
Turtle Library for Generating GCODE

Extruder Turtle Library



Turtle generates a 3D printed path as it moves by generating g-code

https://handandmachine.org/projects/extruder_turtle_rhino/

Thank you!

CS 491 and 591
Professor: Leah Buechley
https://handandmachine.cs.unm.edu/classes/Computational_Fabrication